(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 March 2002 (14.03.2002)

PCT

(10) International Publication Number
WO 02/21761 A2

(51) International Patent Classification[7]: H04L 9/00

(21) International Application Number: PCT/US01/27518

(22) International Filing Date:
6 September 2001 (06.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/656,166 6 September 2000 (06.09.2000) US

(71) Applicant: **WIDEVINE TECHNOLOGIES, INC.** [US/US]; 1301 5th Avenue, Suite 1300, Seattle, WA 98101 (US).

(72) Inventors: **KOLLMYER, Brad**; 849 South Pine Court, Lynden, WA 98264 (US). **BAKER, Brian**; 14631 NE 3rd Street, #7, Bellevue, WA 98007 (US). **SHAPIRO, Eric**; 640 Avis Drive, Suite 300, Ann Arbor, MI 48108 (US). **KOLLMYER, Aric**; 9032 East Shorewood Drive #323, Mercer Island, WA 98040 (US). **RUTMAN, Mike**; 5825 NW 53rd Court, Gainesville, FL 32653 (US). **MACLEAN, Duncan**; 2807 3rd Street, #2, Santa Monica, CA 90405 (US). **ROBERTSON, Dan**; 2595 Powell Avenue, Ann Arbor, MI 48104 (US). **TAYLOR, Neal**; 18361 Avolinda Drive, Yorba Linda, CA 92686 (US). **HUNSCHE, Dick**; 2521 Emerald Avenue, Ann Arbor, MI 48104 (US). **WALKER, Amanda**; 2111 Owls Cove Lane, Reston, VA 20191 (US).

(74) Agents: **BARUFKA, Jack, S.** et al.; Pillsbury Winthrop LLP, 1600 Tysons Boulevard, McLean, VA 22102 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/21761 A2

(54) Title: APPARATUS, SYSTEM AND METHOD FOR SELECTIVELY ENCRYPTING DIFFERENT PORTIONS OF DATA SENT OVER A NETWORK

(57) Abstract: An apparatus and method for selectively encrypting portions of data sent over a network between a server and a client. The apparatus includes parsing means for separating a first portion of the data from a second portion of the data, encrypting means for encrypting only of the first portion of the data, and combining means for combining the encrypted first portion of the data with the second portion of the data, wherein the second portion of the data is not encrypted. The apparatus further includes decrypting means installed at the client for decrypting the encrypted portion of the data. The apparatus is platform independent in terms of media format and data protocol. The encryption unit encrypts data transparently to the client based on the media format. The apparatus of the invention is implemented as one of an application and a plug-in object. The method for selectively encrypting portions of data which differ from each other in at least on characteristic sent over a network between a server and a client includes parsing the data into a first and second portion, encrypting only the first portion of the data, and sending the encrypted first portion and the second portion of the data over the network to the client. The method further includes receiving data from the server, determining whether a data stream is established between the server and the client, and negotiating an encryption key with a decryption shim of the client.

APPARATUS, SYSTEM AND METHOD FOR SELECTIVELY ENCRYPTING DIFFERENT PORTIONS OF DATA SENT OVER A NETWORK

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to transmission of data over a network and, more specifically, to an apparatus, system and method for selectively encrypting and decrypting different portions of data sent over a network.

2. Description of Related Art

Networking, today exemplified by the Internet, began as a way for users to share text-based information and files. The technology has, however, advanced far beyond text. The Internet can be used to conduct videoconferences where the participants in the conference can see each other in real-time on their computer screens or network-connected conferencing device. Internet users can watch live video footage of events as they happen, or view pre-recorded programming "on demand," i.e., whenever they choose rather than on a broadcast schedule.

To understand how video and multimedia content is transmitted over the Internet, first it is necessary to understand the concept of streaming. Streaming solves a long-standing problem inherent in transmitting media information across the Internet: multimedia files are quite large, and most users have relatively low-bandwidth network connections. Sending broadcast-quality video or CD-quality audio across the Internet has never been easy or practical. It could take hours to send a single audio or video file across the network to someone's computer. The person at the other end would have to wait until the entire file was downloaded before playback could begin. Playback might last only a few minutes.

It has long been possible to reduce the amount of data required to transmit multimedia material. One way is to reduce the resolution, frame rate or sampling rate of the transmitted data, which has the side effect of reducing the perceived quality of the audio or video content. Another commonly used method is to apply data compression to the data to remove redundancies and retain only the most essential data from a multimedia file. By combining these two techniques, it is possible to produce multimedia files of reasonable quality that can

be transmitted over typical Internet connections in time frames similar to the running time of the media content itself, i.e. minutes instead of hours. The actual quality of downloaded multimedia material varies depending on the available network bandwidth and the willingness of the user to wait for it to arrive; it is possible to trade off quality against file size and to prepare higher-quality versions of the content for users with more bandwidth.

When the amount of data per second of multimedia content is equal to or less than the amount of bandwidth available to a network user, streaming can occur. Streaming is essentially "just-in-time" delivery of multimedia data. The user does not need to wait for the entire file to arrive before s/he can begin viewing it, nor does s/he need to dedicate a large piece of his/her computer's storage to the received data. As it arrives at the client computer, the multimedia data is typically stored in a buffer capable of holding several seconds' worth of data, in order to avoid interruptions in playback caused by erratic network connections. When the buffer is initially filled, which happens in seconds rather than minutes or hours, playback begins, and individual frames of video and/or audio segments are retrieved from the buffer, decoded, and displayed on the client's video monitor, played through the client's sound system, or otherwise played back via the client's network connected device. After the data has been played, it is in many cases discarded and must be transmitted again if the user wishes to review a previous portion of the program.

The World Wide Web ("Web") operates on a client/server model; that is, a user (i.e. a client) runs a piece of software on his/her personal computer or other network device (such as a network appliance or Internet capable wireless phone) capable of accessing the resources of a network server. The server can allow many different users to access its resources at the same time and need not be dedicated to providing resources to a single user. In this model, the client software-e.g. a browser or player-runs on the user's computer. When a user requests a video or other multimedia content on the Web, his/her client software contacts the Web server containing the desired information or resources and sends a request message. The Web server locates and sends the requested information to the browser or player, which displays the results by interpreting the received data as appropriate, e.g. displaying it as video on the computer's display.

Information on the Web is addressed by means of Uniform Resource Locators (URLs). A URL specifies the protocol to be used to access the required information (commonly HTTP, the HyperText Transfer Protocol), the address or Internet name of the host containing the information, any authentication information required to gain access to the information (such as a User ID and Password), an optional TCP/IP port number if the

resource is not available on the standard port for the specified protocol, and a path to the desired information in a virtual hierarchy designated by the server operators. Additionally, the URL can contain information entered by the user such as query text for a database search. URLs can be specified in many ways, including but not limited to typing them into a Web client, by clicking a "hyperlink" in some other Web document, by choosing a "bookmark" in a Web browser to return to a previously-visited page, or by filling out a Web form and submitting it to the server.

When a request is made for a particular URL, the request is sent to the appropriate server using the protocol indicated in the URL, such as HTTP, and the requested data (or an error message if the request cannot be fulfilled) is returned by the same method. However, HTTP and other Web protocols are built on top of fundamental Internet protocols, such as TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). The protocols used for streaming media are built on top of these fundamental protocols in the same way that other high-level protocols, such as HTTP, are. It is essential to realize that all Internet traffic is carried by a relatively small number of low-level, fundamental protocols such as TCP and UDP, which are used to establish the connection between computers that carries the higher-level protocol. When streaming multimedia content, then, the server may use either TCP or UDP.

RTP (Real-Time Protocol), RTSP (Real-Time Streaming Protocol), RTCP (Real-Time Control Protocol) are three of the many possible streaming media protocols that can be built on top of the Internet's low-level protocols. In fact, RTP, RTSP, and RTCP serve complimentary functions and are most often used together as part of some of the more common streaming implementations. RTSP is used to set up and manage a streaming connection to between a server and a client, RTP is used to deliver the actual multimedia data, and RTCP provides timing and other control signals between the client and the streaming server. Often, RTP is sent on top of UDP, a low-overhead protocol that delivers the data as quickly as possible, but does not guarantee that any particular piece of data (a "packet" in network terminology) will actually arrive at its destination. RTCP packets are sent interleaved with the data or in parallel with the RTP media channel via TCP.

One serious drawback of using the Internet for the distribution of streaming media is that the data of necessity often passes through networks and systems that are not controlled by the sender of the data en route to the client. Once the data leaves the sender's protected network, it is vulnerable to interception. This is of particular concern when proprietary data, such as a motion picture, is transmitted across the network, be it the public Internet or a third

party private network. The data can potentially be intercepted and copied at any point as it is transmitted across these networks. Without some way to protect data from interception and unauthorized duplication, the Internet will never provide the security necessary to allow copyright owners to safely distribute their works over the network.

Encryption is one way to address this issue. Encryption provides a way to encode information such that only the intended recipient can view it. While anyone can intercept the data, only the legitimate recipient will be able to decrypt it, retrieve the original message or media content, and display it. Many encryption solutions have been created to provide this type of security. For example, software referred to as VPN (Virtual Private Network) allows a group of computers connected to the Internet to behave as if they were connected to a physically secure local network, using encryption to ensure that only computers on the VPN can access the private network resources. Other applications of encryption to allow private communications over the Internet include but are not limited to PGP (Pretty Good Privacy), IPSec (EP Security) and SSL (Secure Sockets Layer).

Organizations, often but not limited to corporations, also protect their proprietary data by use of firewalls. Every time a corporation connects its internal computer network or local area network (LAN) to the Internet, it faces a similar problem of private data being intercepted. Whereas encryption is used to ensure that data sent through the public Internet is usable only by intended recipients, firewalls are aimed at keeping proprietary information secure on a LAN which is connected to the Internet by preventing unauthorized users from accessing information stored on the internal network via the public Internet. Due to the Internet's public nature, every LAN connected to it is vulnerable to attack from the outside. Firewalls allow anyone on the protected LAN to access the Internet in ways allowed by the firewall configuration, while stopping hackers on the Internet from gaining access to the LAN and stealing information and/or deleting or otherwise vandalizing valuable data.

Firewalls are special-purpose devices built on routers, servers, and specialized software. One of the simplest kinds of firewalls uses packet filtering. In packet filtering, a router screens each packet of data traveling between the Internet and the LAN by examining its header. Every TCP/IP packet has a header containing the IP address of the sender and receiver as well as the port number of the connection and other information. By examining the header, and particularly the port number, the router can determine with a fair amount of accuracy the type of Internet service each packet is being used for. Each Internet service, including HTTP (the Web), FTP (File Transfer Protocol), Telnet, rlogin, and many others, has a standard port number that is used by convention for most access to the service. While

any port number can be used for any service, it is far more convenient for users to use the standard port numbers, and virtually all Internet services intended for access by the public do so. Once the router knows what client and what service a packet is intended for, it can simply block outside access to hosts and services that public Internet users should not be able to use. System administrators set the rules for determining which packets should be allowed into the network and which should be blocked.

Proxy servers are commonly used in conjunction with firewalls. A proxy server is a software gateway that runs on a computer that is accessible by both the protected LAN and the Internet. All access to the Internet from the LAN must go through the proxy server, as must all access to the LAN from the public Internet. When a computer on the LAN requests an Internet resource such as a Web page, that request is sent to the proxy server. The proxy server then makes the request to the Internet resource and forwards any returned data back to the original requester on the LAN. Since the Internet and the LAN touch only at the single point of the proxy server, which acts as a go-between, protecting the network involves securing only one computer, rather than dozens or hundreds. Proxy servers are frequently used by employers to control and monitor how their employees use the Internet, as well as for preventing intrusion attempts from the Internet.

NAT, the term used for networks that utilize Network Address Translation, complicates delivery of Internet data as well. In a NAT, a privately addressed network is established separated from the Internet by a NAT router. This router in turn has an address on the public Internet. By translating the addresses of the private network into addresses recognizable on the public Internet and vice versa, the NAT router facilitates Internet connectivity for the privately networked machines. NATs are often used where limited numbers of public addresses are available in comparison to the number of users (a user might use a NAT for multiple machines sharing a cable modem connection) or in conjunction with firewalls and/or proxies to provide an extra layer of security to the private network.

Data packets can be typically divided into two parts, the header and the payload parts. The header is the portion of the packet that includes routings or other configuration information. The payload is the portion of the data packet that is just the data of interest, in exemplary case: multimedia content. For example, in a network packet, the header contains data for use by network routers in delivering the packet to its final destination, as well as other data about the packet such as size and formatting information. In an exemplary RTP packet, the header contains channel information as well as other information needed by the player to direct the RTP (media content) payload. Some complex packets may contain

multiple headers and diverse nonpayload information and will be referred to herein as the non-payload part.

Many current encryption solutions, such as IPsec, encrypt data far too indiscriminately, encrypting not only the payload but portions of the header or nonpayload part as well. Only the routing information remains unencrypted, thereby rendering information needed for firewalls, proxies and/or NATs to appropriately relay the packets scrambled which will stop the packet before it can transit into a private or protected network. Other solutions provide inadequate encryption to assure integrity of the data across the public network. Some streaming media encryption solutions deal with the problem by placing the encryption system on the server machine in the form of special software that encrypts the streaming media before it is converted to packets for network transmission. While this results in packets that can often successfully pass through proxy servers and firewalls, this strategy is limiting. This is because this solution requires modification of the streaming server and often requires the addition of extra streaming servers or processing capacity to handle the increased workload that encryption adds. Also, the solution must be reengineered for each streaming server platform supported.

BRIEF SUMMARY OF THE INVENTION

The present invention solves the above and other problems by providing an apparatus, system and method that parse and selectively encrypt different portions of the data in real-time, decrypt the encrypted data in real-time and pass the data to a media player on a client computer or other network capable device. In particular, there is provided an encryption bridge that examines, parses and selectively encrypts only the payload (e.g. Media content) portion of the data, leaving the non-payload portion intact such that the data can cross firewalls, proxies and NATs without the firewalls, proxies or NATs having to be modified to accommodate the encrypted data. The invention ensures that it does not encrypt the portion of the data stream that firewalls, proxies, and NATs require to properly deliver the data to the intended recipients (e.g. Client computer). Thus, if the encryption unit does not see a data type that it specifically recognizes, then it ignores it, but if the encryption unit sees a data type that it does recognize (e.g. Multimedia content), then it selectively encrypts only the recognized portion of the data stream.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate at least one exemplary embodiment of the invention and, together with the description, explain the various advantage and principles of the invention. In the drawings:

FIG. 1 is an illustration of an exemplary computer system on which an encryption bridge may be implemented.

FIG. 2 is a flowchart of an exemplary encryption process.

FIG. 3 is a flowchart of an exemplary decryption process.

FIG. 4 is an exemplary diagram of a shim used in a Windows™ Sockets Network Architecture.

FIG. 5 is an exemplary diagram of a shim used in a Streams-Based Network Architecture.

DETAILED DESCRIPTION OF THE INVENTION

The following detailed description refers to the accompanying drawing, which illustrates the exemplary embodiment of the present invention. Other embodiments are possible and modifications may be made to the exemplary embodiments without departing from the spirit and scope of the invention. Therefore, the following detailed description is not meant to limit the invention. Rather, the scope of the invention is defined by the appended claims.

FIG. 1 illustrates an exemplary computer system 100 which includes an encryption bridge (also referred to as "EB") 110 of the invention interposed between the streaming servers 120, which provide media streams (payload data) 170, and an outside network such as the Internet 130, and a user or client computer 140. The EB 110 operates such that as data is sent through EB 110, only selected portion of the data (e.g., multimedia data) are encrypted. The EB 110 can be set up such that it encrypts other data as well. For example, the EB 110 can be configured to encrypt/protect data that might include but is not limited to electronic books, medical records or images, financial data or any other data that requires secure transmission over a network.

The EB 110, which may represent a single machine or a cluster of machines, is interposed between servers 120 and the Internet 130. All pertinent data moving from the servers 120 and the Internet 130 via the EB 110 needs to be made secure by encryption;

traffic between the EB 110 and the servers 120, however, does not have to be encrypted since it is in a "safe zone" by virtue of not being exposed to the outside network. Data channels 150 may represent control data such as pause, play and rewind or speed modulation data for the streaming servers 120 or can be used for more complex data forms and monitoring. Additionally, the data channels 150 can be used to hand this monitoring data off to other machines such as e-commerce or quality monitoring servers for services not directly related to the stream encryption process.

The EB 110 passes data to network clients 140 and ensures that the data owners or content providers (e.g. Yahoo, Inc., AOL, Inc., etc.) that the data from their streaming servers 120 will be securely encrypted while traversing the Internet 130 and that once on the client machine 140 the data will be difficult to copy. Security on the client machine 140 can be accomplished by a variety of techniques, including but not limited to checking for and preventing known piracy techniques and monitoring the client machine 140 for suspicious behavior. The streaming servers 120 and the clients 140 are the ultimate sources and destinations, respectively, of the data streams (e.g. multimedia content) 170 and data channels (e.g. control data) 150 transported via the EB 110. An additional data handler (e.g. e-commerce system, quality monitoring system, etc.) 160 may also be connected to the EB 110, which is further described below.

The streaming server 120 operators (e.g. content owners or service providers to content owners) are the system's primary customers, which provide the content to transmissions requested by a client 140 over a network such as the Internet 130. The client 140, for the purpose of this example, is equipped with media playback software. For this example, the media playback software could be the Apple Computer Quicktime™ Player, Real Player™, the Windows Media Player™ or any other software capable of playing back multimedia streams. The Internet 130 is a public network. The data handler 160 is, for example, an e-commerce system or other outside monitoring or data system that a customer may choose to utilize in conjunction with a streaming security solution.

FIG. 2 is a flowchart of an exemplary encryption process of the invention for parsing and selectively encrypting data being sent over a public network between a server and a client. This process comprises recognizing a new data stream arriving at the EB 210; determining whether the data is to be encrypted 220; ignoring and passing the data stream if it is not to be encrypted 222; determining whether a shim is present on the client/target 230; deploying the shim if it is not already present 232; determining whether an encryption key is

current 240; exchanging encryption key(s) with the client side shim if the key was not current 242; parsing data into payload and non-payload parts 250; encrypting the payload part using the exchanged key 260; passing the data consisting of payload and non-payload parts across the network 270; determining whether the processed data is last in the stream 280; receiving feedback from the shim if the data was not the final part of the stream 282; determining if the client is compromised 284 (e.g. is hacked or the data needing security is otherwise placed at risk); shutting down the data stream if the client is compromised 286; resuming the parsing of the established stream if the client is not compromised 288; and terminating the streaming session if the data was the last in the stream 290.

Benefits of the EB system include: the EB fits beside the customers' streaming server system with minimal integration costs; there is no need to modify the server system for the EB to work; the EB is transparent to both client side software and the streaming server system with client software appearing to communicate with the servers without an unexpected intermediary and servers delivering data to clients without detecting the ongoing encryption process.

In addition, the EB system provides parsing and encryption for a variety of data types and streams including but not limited to Windows Media™ format and RTP/RTSP using TCP, TCP/UDP or http delivery. All payload data sent via these formats are encrypted as the data travels from the server through the EB to the client.

Encryption is accomplished via an encryption algorithm such as DES-X, a technique well known in the art, and other algorithms can be substituted depending on customer and other needs. The parsing and encryption of the data streams does not interfere with the stream's ability to pass through proxy server, firewalls or NATs that parse packets to facilitate delivery of data to a protected LAN. This is an important feature of the invention since the applicants are not aware of any software that parses for content on a selective basis and subsequently acts on just the payload part of the data like the present invention. Moreover, outside authentication is not required since the EB handles exchange of encryption keys directly with the client.

In addition, the EB is scalable to allow the addition of more encryption machines as the bandwidth increases. The encryption machines are scalable to allow the addition of additional processor and network cards to handle increased bandwidth. This allows the system maximum growth with minimal increase in the physical size of the EB.

As more encryption machines are added to the EB, the encryption is spread out among those available encryption machines to maximize throughput. The EB can be configured while it is online to: add or remove encryption machines; shutdown the EB; prevent the system from accepting new streams and allow existing streams to run out; read status to tell that a given encryption machine is no longer servicing streams and can be safely removed from the system; and gather data on streaming capacity, usage and quality of data transferred via the EB.

In addition, the EB and shim architecture provide pluggable cores for changing and maintaining features such as the encryption algorithm and client side security monitors. A pluggable core is modular code that allows portions of the whole program to be readily replaced without disturbing the remaining functionality. For example, the invention allows the EB to encrypt with a variety of algorithms such that when a new data stream is established through the EB, a new encryption core could be sent to the client to allow the payload part to be encrypted with a different algorithm (e.g. Blowfish or RSA™ instead of DES-X). Stated another way, the encryption algorithm can be changed at any time since the software architecture is entirely pluggable/exchangeable. As to the key exchange system, though currently standard protocols are used, they can be changed at any time since this architecture is also pluggable/exchangeable. As to the client side security system (i.e. hacker detection), this too can be altered at any time since the architecture is pluggable/ exchangeable. All core functionality of the client side shim is implemented via this pluggable/exchangeable architecture.

Another novel aspect of the invention is the "on-the-fly" encryption, which makes data packets that are transmitted over the network useless to any computer other than the intended machine. This is done transparently relative to the client media playback mechanism (e.g. media player software) and server and in real-time. The initial implementation of the invention utilizes DES-X, a speed optimized algorithm well-known in the art, though via the pluggable core architecture noted above this could be changed out for other algorithms that may include but are not limited to RSA™ algorithms or the widely known Blowfish and Two-Fish algorithms.

Thus, the invention provides a software bridge that examines network data passing through, parses the network data and only encrypts the relevant payload part, leaving the non-payload part that may include data such as routing, size and other header data surrounding the payload part entirely untouched. In other words, certain portions of the data

are encrypted and other portions of the data are not or do not need to be encrypted. What the systems and methods of the invention do is parse the network and actually look at the data format, encrypting only the portion of the data that contains, in the exemplary case, multimedia content. Selectivity of the data to be encrypted is based on the format of the data sent, which the EB recognizes and responds to appropriately. The EB of the invention sits in between the servers and clients and encrypts the data. Decryption takes place on the client using the shim and pluggable cores as noted above. This way, the data if intercepted on the network is encrypted and useless to the interceptor. The data cannot be intercepted prior to encryption since the EB and servers are connected via a secure network connection.

As stated above, the encrypted data stream of the invention can pass through a proxy server, firewall or NAT without those mechanisms needing to be modified to accommodate the data stream. A benefit of the invention over current encryption solutions is the ability to produce encrypted data streams that can cross through proxy servers, firewalls and NATs. This ability to produce data that crosses unmodified proxy servers, firewalls and NATs results from the invention's ability to selectively parse data into payload and non-payload parts and encrypt only the payload part, a level of selectivity that current encryption solutions do not provide.

For the streaming server operators this becomes very significant since many home, office and other networks are protected or isolated from the public Internet by firewalls, proxy servers and/or NATs. With current encryption solutions that encrypt data less discriminately, the data is unable to be delivered across unmodified firewalls, proxy servers and NATs.

For instance, when a user requests data from a streaming server, that data stream is organized into packets that have specific data for identifying the target user. Without accurately parsing the data into payload and non-payload parts, the user specific data is readily damaged or scrambled during the encryption process, making it impossible for the firewall, proxy server or NAT to deliver the data to the requesting user. In contrast, the present invention accurately separates the payload and non-payload parts, encrypting only the payload part so that the data appears unchanged to the firewall, proxy server or NAT that requires only the non-payload part to affect delivery to the user requesting the data stream.

A firewall, for example, does not recognize or try to block the encrypted data stream because the transport protocols do not define the appearance of the payload part, only the appearance of the non-payload part. The firewall looks at the nonpayload part including but not limited to size, routing and header data. If the nonpayload part data identify the data

stream as a reply to a user request, then firewall determines that the data stream is not malicious in origin and will not prevent it from going through. However, if the firewall is unable to parse the non-payload part or does not recognize the non-payload part than the data will be blocked from passing through.

In existing encryption solutions where the entire data portion of packets is encrypted, special modifications in each firewall, proxy server or NAT that the data stream might pass through are necessary. That is, the firewalls, proxy servers and NATs would have to be updated to identify the encrypted data. The present invention does not require modifications of the firewalls, proxy servers or NATs already deployed because it selectively encrypts the data packets leaving the portions important to firewalls, proxy servers and NATs unchanged such that the firewall, proxy server or NAT can pass the data stream to the intended target.

Some existing encryption solutions exist that encrypt only the media portion of a data stream by placing the encryption software on the streaming server as a plugin to streaming server software, placing a heavy processing burden on the streaming server. This is in contrast to a benefit of the invention in that the invention can be used with a plurality of streaming servers without modification being required to the streaming servers and providing encryption without impacting the processing performance of the streaming servers.

Another feature of the invention is it provides a system that is independent in terms of the media format used. That is, the invention operates based on data protocol rather than file format. Multimedia streaming over networks is accomplished via several protocols. The invention recognizes the streaming protocol and acts on the data rather than requiring specific identification of the file format being transmitted. The invention is also independent in terms of the operating systems on the server machines since the invention requires no direct access to the server machines, the invention merely requires that the data streams from the streaming server pass through the EB.

The invention further provides a client system, also referred to as a decryption shim or simply a shim, which is a piece of transparent software that is downloaded to or pre-installed on the client machine (e.g. personal computer, network appliance or other network capable device) and used to decrypt incoming data streams from the EB on its way to the media player software. FIG. 3 is a flowchart of an exemplary decryption process of the decryption shim software performed at the client machine. The process comprises data streams as they are initiated 310; determining whether the data is an encrypted stream 320; ignoring the stream if it is not encrypted data 322; determining if the encryption key is current 330; negotiating a key with the encryption bridge/source if the key is not current 340; parsing the

data into payload and non-payload parts 350; decrypting only the payload part 360; passing the data to higher level operations (e.g. the media player) 370; determining whether the data is the last part of a stream 380; examining the operating environment for security 382; determining if the client environment is compromised (hacked, etc.) 384; shutting down the data stream if the client is compromised 385; communicating with the encryption bridge/ source 386; resuming parsing data 388; and terminating the stream if the data was the last part of the stream 390.

Decryption is accomplished by adding a decryption shim 420 in a Layered Service Provider 410 in a Windows™ sockets network architecture as shown in FIG. 4 or a streams plug-in 510 in a streams based network architecture as shown in FIG. 5. FIG. 4 is an exemplary diagram of a Windows™ sockets network architecture. In the Windows™ networking architecture, decryption shim 420 is the highest most Layered Service Provider (LSP) so that an additional LSP cannot merely spool decrypted data out into an insecure environment. This can be extrapolated to other sockets based networking protocols as well. FIG. 5 is an exemplary diagram of a streams based network architecture. The diagram represents placement of the invention's shim 520 within a streams based architecture 510 such as that employed by current incarnations of Mac OS™ and some versions of Unix.

When a user requests data that is encrypted by the EB of the invention, the transparent software is installed via an Active-X™ Control, a well documented means to deliver executable programs to a Windows™ computer. The installation of the decryption shim is transparent to the user and does not cause a reboot, restart of the user's browser or require user interaction. Some exceptions such as the Mac OS™ and Windows NT™ or Windows 2000™ in secure environments or Linux or Unix based client machines because transparent installation requires administrative user privileges on the client machine and the ability of the client machine to receive programs via the Active-X™ mechanism.

After the last stream has finished, the decryption shim uninstalls as much of itself as possible, leaving only a small layer so that administrative user privileges are not required for future decryptions. The decryption shim is installed in volatile memory to reduce the changes of tampering by a third party.

After installation, the decryption shim decrypts only the data coming from the EB of the invention going to targeted players such as Windows Media™ Player, QuickTime™ Movie Player, Real Player™, etc. Decryption does not impact data targeted to other applications or media streams that are not encrypted by the EB of the invention.

The decryption shim runs on, for example, operating systems such as Windows 95™, Windows 98™, Windows ME™, Windows NT™, Windows 2000™ as well as Mac OS™ and numerous Linux and Unix distributions. Where Active-X™ based installation is possible, the installation of the decryption shim can be accomplished with most browsers such as Internet Explorer™ or Netscape™.

In sum, the EB of the invention drops in between the server and client and parses and encrypts a selected portion of the streamed data such as the media portion. As the stream is initiated, the decryption core is sent as part of the stream to the client side. The client can then decrypt the incoming data for the duration of the stream. If another stream is initiated, decryption occurs the same way. For each stream, the encryption keys can be set for the duration of the stream or changed during the stream duration to increase security.

As an example, a client may request privileges to get streaming data from a service provider's e-commerce system. The service provider's back-end (server infrastructure) will authorize the streaming server to initiate a stream. That stream is initiated through the EB. Once initiated, the stream is parsed and selectively encrypted by the EB before being passed out over the network. Encryption keys are exchanged, for example, by the Diffy-Helman mechanism that is known in the field. Unique features of the invention include the parsing and selective encryption of only the payload part of the data stream and the ability to plug-in other key exchange mechanisms and encryption algorithms should customer or security needs dictate.

It will be apparent to one of ordinary skill in the art that the embodiments as described above may be implemented in many different embodiments of software, firmware and hardware in the entities illustrated in the figures. The actual software code or specialized control hardware used to implement the present invention is not limiting of the present invention. Thus, the operation and behavior of the embodiments were described without specific reference to the specific software code or specialized hardware components, it being understood that a person of ordinary skill in the art would be able to design software and control hardware to implement the embodiments based on the description herein.

CLAIMS

1. An apparatus for selectively encrypting data for transmission over a network between a server and a client, the apparatus comprising:

means for parsing a first portion of the data from a second portion of the data;

means for encrypting only the first portion of the data; and

means for combining the encrypted first portion of the data with the second portion of the data,

wherein the second portion of the data includes more than routing information.

2. The apparatus of claim 1, wherein the data includes streaming data.

3. The apparatus of claim 1, wherein the first portion of the data includes payload data.

4. The apparatus of claim 1, wherein the second portion of the data includes at least one of a header, control data and routing data.

5. The apparatus of claim 1, further comprising means for sending the combined first and second portions of the data over the network to the client.

6. The apparatus of claim 1, further comprising means for receiving the data from the server before the data is sent over the network to the client.

7. The apparatus of claim 1, further comprising means for establishing a data stream between the server and the client.

8. The apparatus of claim 1, further comprising key-negotiating means for negotiating an encryption key with the client.

9. The apparatus of claim 8, wherein key negotiation and key exchange occur during transmission of a stream.

10. The apparatus of claim 9, wherein encryption by the encrypting means is transparent to the server.

11. The apparatus of claim 8, wherein key negotiation can determine the correctness of the result.

12. The apparatus of claim 1, further comprising decrypting means installed at the client for decrypting the first portion of the data.

13. The apparatus of claim 1, wherein the parsing means parses the data into different portions based on media format.

14. The apparatus of claim 1 wherein the encrypting means encrypts the first portion of the data based on media format.

15. The apparatus of claim 1, wherein the apparatus is implemented utilizing an application that includes a pluggable core encoding an encryption algorithm for encrypting the first portion of the data, wherein the pluggable core enables the encryption algorithm to be readily changed.

16. The apparatus of claim 1, wherein the apparatus in implemented on a encryption bridge.

17. A method for selectively encrypting data received from a data source, the data including first and second portions which differ from each other in at least one characteristic, the received data to be subsequently sent over a network to a client, the method comprising:

parsing the received data into portions including the first and second portions;

encrypting the first portion of the received data; and

sending the received data including the encrypted first portion and the second portion of the received data over the network to the client.

18. The method of claim 17, wherein the data source is a server.

19. The method of claim 17, further comprising determining whether a stream is established between the server and the client.

20. The method of claim 15, further comprising negotiating an encryption key with the client.

21. The method of claim 20, wherein the received data from the data source is streaming data sent during a streaming session and the negotiating of the encryption key is carried out during the streaming session.

22. The method of claim 20, wherein the received data from the data source is streaming data sent during a streaming session, the method further comprising examining the client during the streaming session and terminating the streaming session if the encryption key on the client is invalid.

23. The method of claim 20, wherein the encryption key is negotiated with a decryption shim on the client.

24. The method of claim 17, further comprising determining whether the received data is streaming data.

25. The method of claim 24, further comprising parsing, encypting and sending the data if the data is streaming data and sending the data if the data is not streaming data.

26. The method of claim 17, further comprising determining whether a shim is present on the client.

27. The method of claim 26, further comprising sending a shim to the client if it is determined that the shim is not present on the client.

28. The method of claim 17, further comprising determining whether an encryption key on the client is current.

29. The method of claim 17, wherein the data includes a payload data portion and at least one of a header, control data and routing data.

30. The method of claim 29, wherein the first portion of the data includes the payload data portion.

31. The method of claim 17, wherein the data received from the data source for sending to the client is a stream of packets, the method further comprising determining whether a packet is the last packet in a data stream.

32. The method of claim 31, further comprising receiving feedback from a decryption shim on the client if it is determined that the packet is not the last packet in the data stream.

33. The method of claim 17, further comprising determining whether the client is compromised.

34. The method of claim 33, further comprising continuing parsing, encrypting and sending the data into the first and second portions if it is determined that the client is not compromised.

35. The method of claim 33, further comprising terminating the sending to the client if it is determined that the client is compromised.

36. A method for decrypting streaming data at a client, the data including first and second portions which differ from each other in at least one characteristic, the data having

been sent over a network to the client from an encryption source, the encryption source having encrypted the first portion of the data, the method comprising:

receiving the data sent over the network;

parsing the data into portions including the first and second portions;

decrypting the first portion of the data; and

passing the decrypted first portion of the data to a higher level of operations for play in the client.

37. The method of claim 36, further comprising prior to the parsing, determining whether the data is an unencrypted stream.

38. The method of claim 37, further comprising passing the data to a higher level of operations without parsing and decrypting when it is determined that the data is an unencrypted stream.

39. The method of claim 36, further comprising negotiating a decryption key with the encryption source.

40. The method of claim 39, wherein the streaming data is sent from the encryption source during a streaming session and said negotiating the decryption key is carried out during the streaming session.

41. The method of claim 39, further comprising terminating the stream if the encryption key is invalid.

42. The method of claim 36, wherein the first portion of the data includes a payload data portion.

43. A method of claim 36, wherein the data is sent from the encryption source over the network as a stream of data packets, the method further comprising determining whether a packet received by the client is a last packet in a data stream.

44. The method of claim 43, further comprising sending feedback to the encryption source if it is determined that the packet is not the last packet in the data stream.

45. The method of claim 36, further comprising determining whether the client is compromised.

46. The method of claim 45, further comprising continuing the parsing, decrypting and passing the data as aforesaid if it is determined that the client is not compromised.

47. The method of claim 45, further comprising terminating a streaming session if it is determined that the client is compromised.

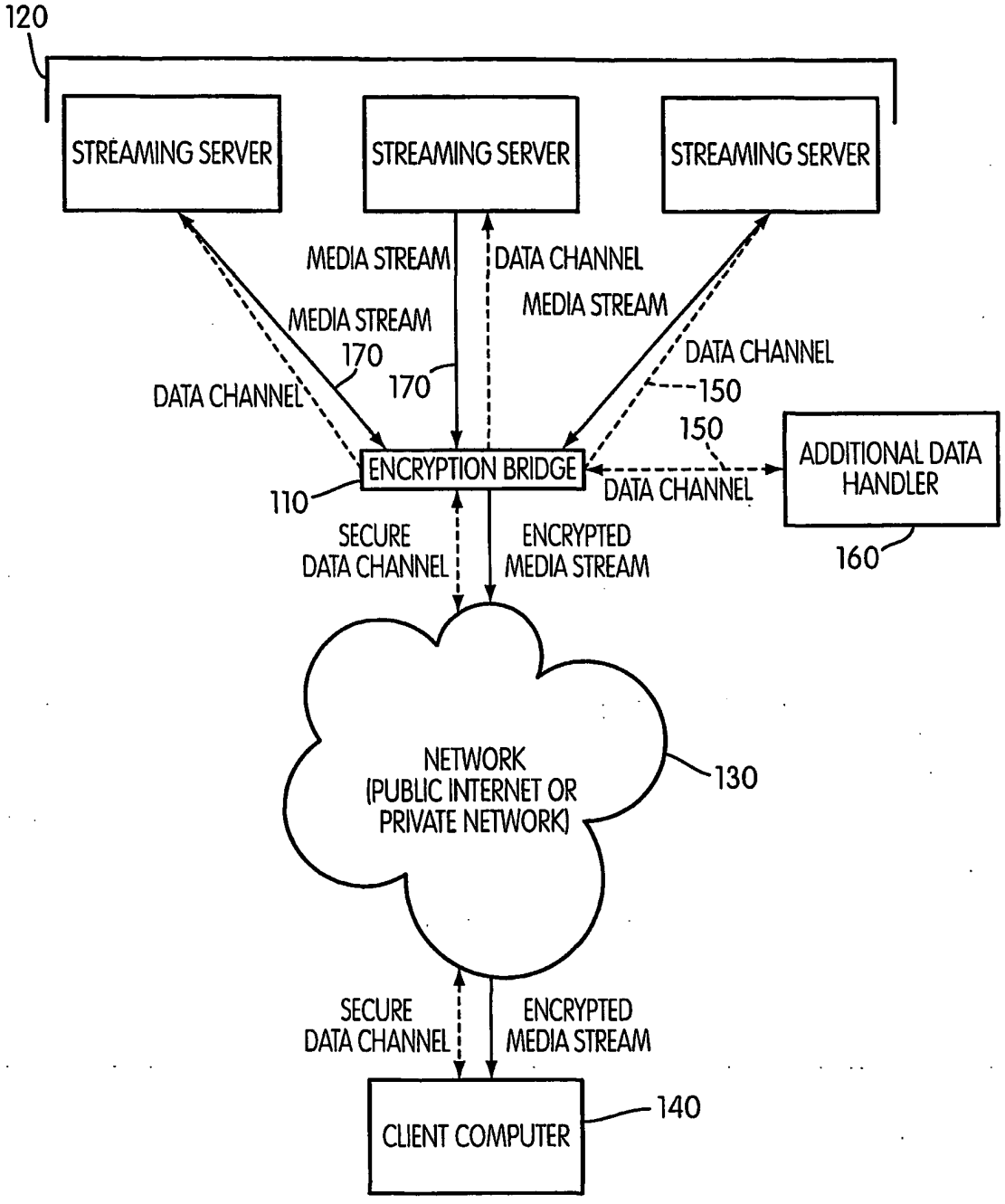
120
STREAMING SERVER
STREAMING SERVER
STREAMING SERVER
MEDIA STREAM
DATA CHANNEL
MEDIA STREAM
MEDIA STREAM
170
170
DATA CHANNEL
DATA CHANNEL
150
150
ENCRYPTION BRIDGE
ADDITIONAL DATA HANDLER
DATA CHANNEL
110
160
SECURE DATA CHANNEL
ENCRYPTED MEDIA STREAM
NETWORK (PUBLIC INTERNET OR PRIVATE NETWORK)
130
SECURE DATA CHANNEL
ENCRYPTED MEDIA STREAM
CLIENT COMPUTER
140

FIG. 1

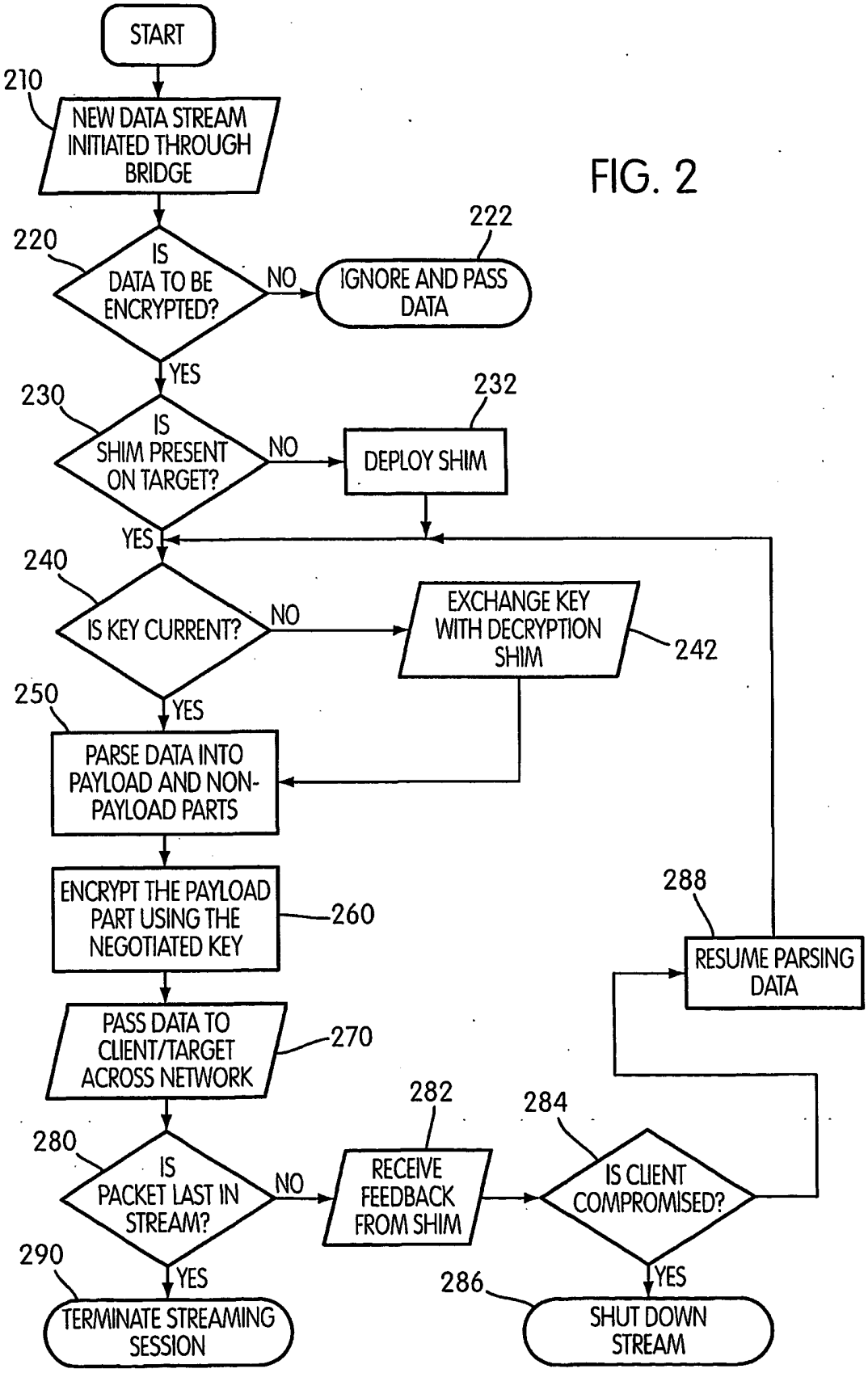
START
210
NEW DATA STREAM INITIATED THROUGH BRIDGE
220
IS DATA TO BE ENCRYPTED?
NO
222
IGNORE AND PASS DATA
YES
230
IS SHIM PRESENT ON TARGET?
NO
232
DEPLOY SHIM
YES
240
IS KEY CURRENT?
NO
EXCHANGE KEY WITH DECRYPTION SHIM
242
YES
250
PARSE DATA INTO PAYLOAD AND NON-PAYLOAD PARTS
ENCRYPT THE PAYLOAD PART USING THE NEGOTIATED KEY
260
PASS DATA TO CLIENT/TARGET ACROSS NETWORK
270
280
IS PACKET LAST IN STREAM?
NO
282
RECEIVE FEEDBACK FROM SHIM
284
IS CLIENT COMPROMISED?
YES
286
SHUT DOWN STREAM
288
RESUME PARSING DATA
290
YES
TERMINATE STREAMING SESSION

FIG. 2

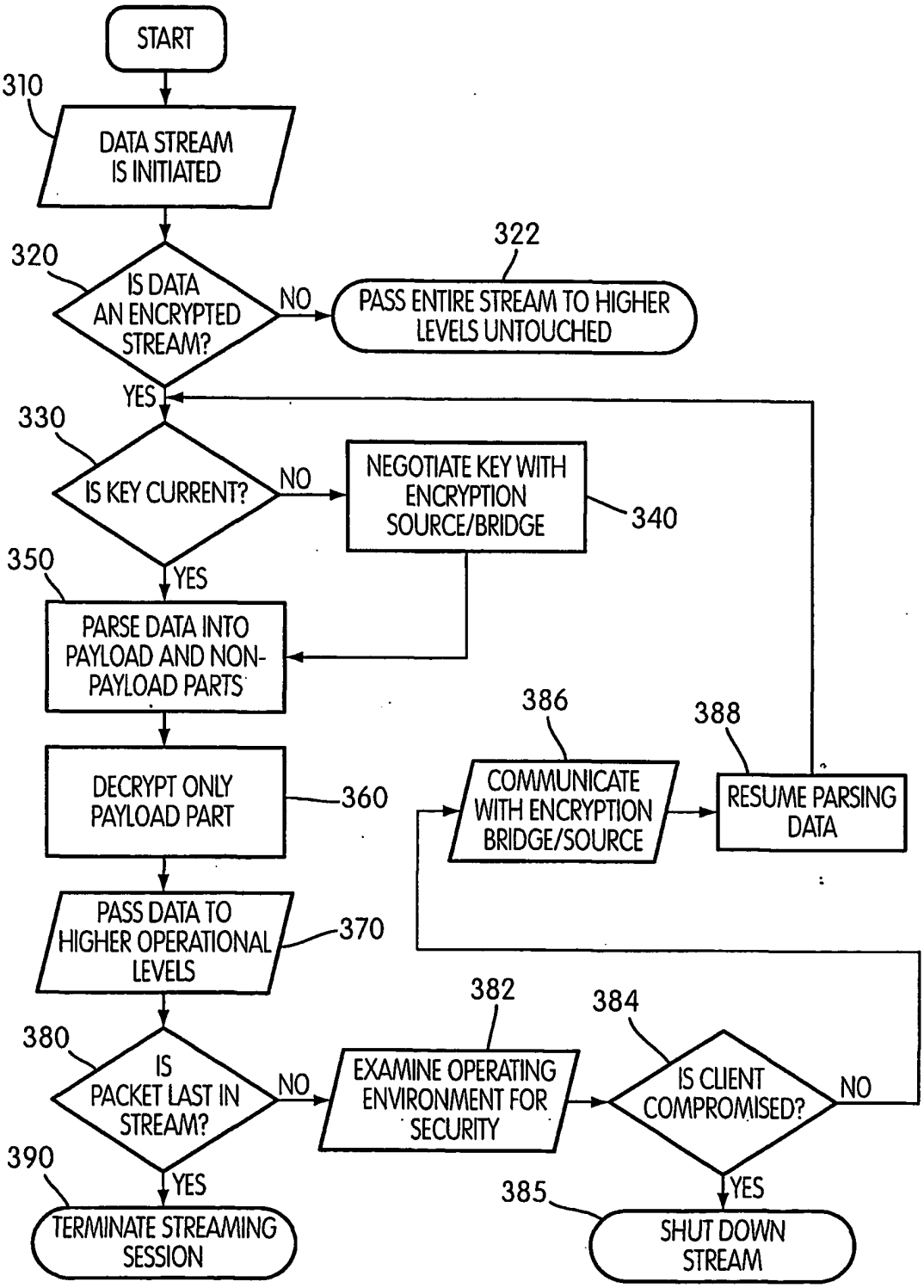
START
310
DATA STREAM IS INITIATED
320
IS DATA AN ENCRYPTED STREAM?
NO
322
PASS ENTIRE STREAM TO HIGHER LEVELS UNTOUCHED
YES
330
IS KEY CURRENT?
NO
NEGOTIATE KEY WITH ENCRYPTION SOURCE/BRIDGE
340
YES
350
PARSE DATA INTO PAYLOAD AND NON-PAYLOAD PARTS
DECRYPT ONLY PAYLOAD PART
360
386
COMMUNICATE WITH ENCRYPTION BRIDGE/SOURCE
388
RESUME PARSING DATA
PASS DATA TO HIGHER OPERATIONAL LEVELS
370
382
384
380
IS PACKET LAST IN STREAM?
NO
EXAMINE OPERATING ENVIRONMENT FOR SECURITY
IS CLIENT COMPROMISED?
NO
390
YES
YES
385
TERMINATE STREAMING SESSION
SHUT DOWN STREAM

FIG. 3

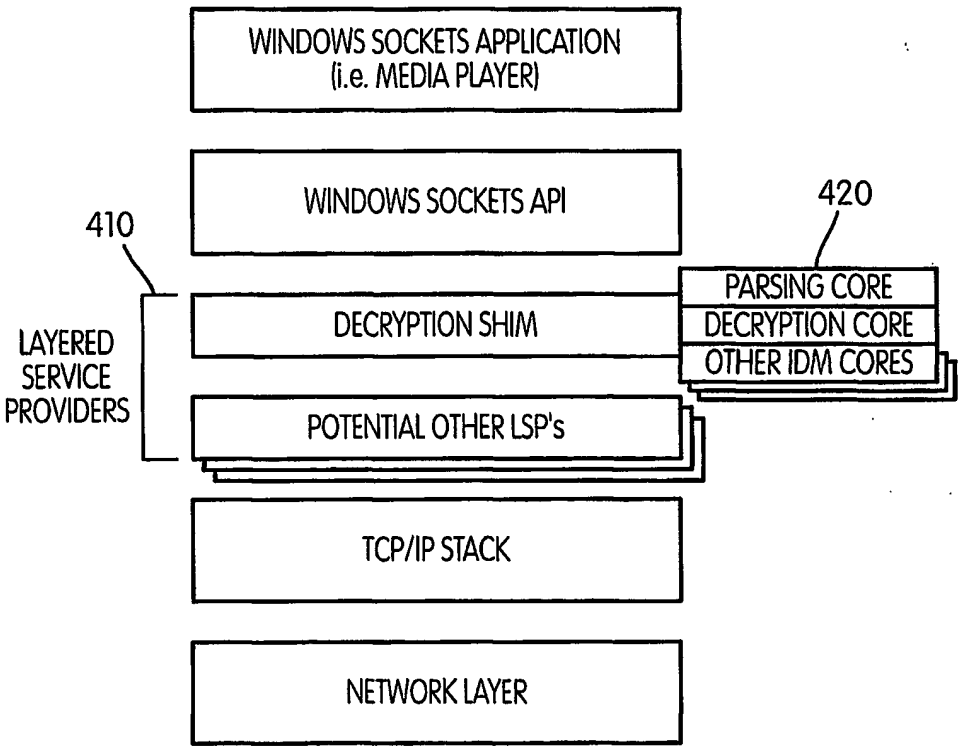
WINDOWS SOCKETS APPLICATION
(i.e. MEDIA PLAYER)
WINDOWS SOCKETS API
410
420
LAYERED
SERVICE
PROVIDERS
DECRYPTION SHIM
PARSING CORE
DECRYPTION CORE
OTHER IDM CORES
POTENTIAL OTHER LSP's
TCP/IP STACK
NETWORK LAYER

FIG. 4

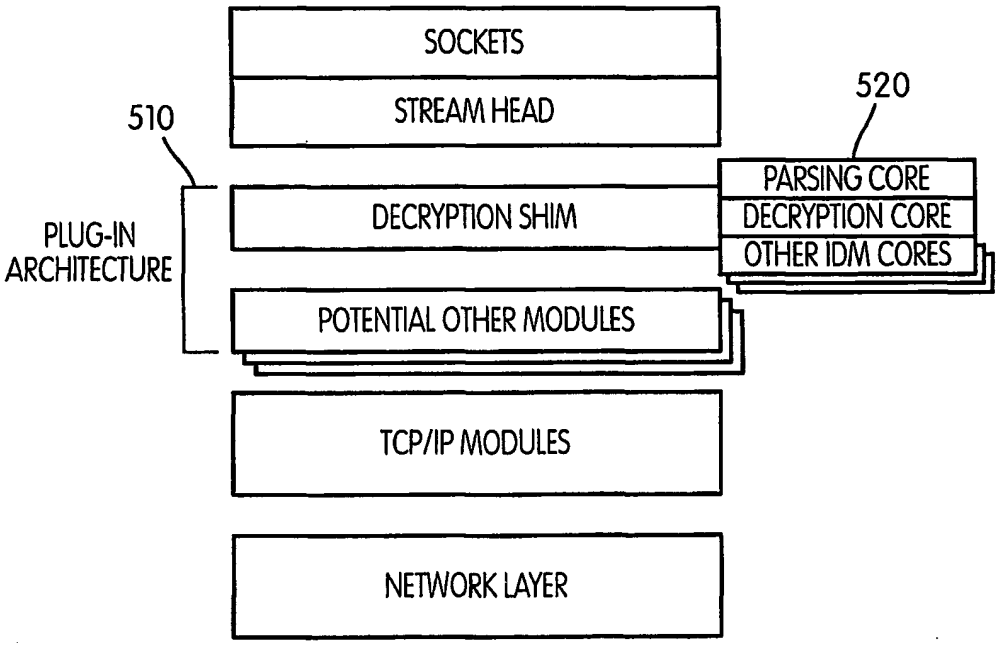
SOCKETS
STREAM HEAD
510
520
PARSING CORE
DECRYPTION SHIM
DECRYPTION CORE
OTHER IDM CORES
PLUG-IN ARCHITECTURE
POTENTIAL OTHER MODULES
TCP/IP MODULES
NETWORK LAYER

FIG. 5